山东大学 计算机科学与技术 学院

操作系统 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201705130113 | 姓名：黄瑞哲 | | 班级：计科17.3 |
| 实验题目：进程控制实验 | | | |
| 实验学时：2 | | 实验日期：2019.10.21 | |
| 实验目的：  加深对于进程并发执行概念的理解。实践并发进/线程的创建和控制方法。观 察和体验进程的动态特性。进一步理解进程生命期期间创建、变换、撤销状态变换 的过程。掌握进程控制的方法，了解父子进程间的控制和协作关系。 | | | |
| 硬件环境：  Intel Core i5-8300H | | | |
| 软件环境：  Windows 10 Pro 1903  Docker Desktop for Windows  Ubuntu 18.04  VS Code | | | |
| 实验内容与设计：   1. 了解相关系统调用的用法   Pid\_t fork()  成功创建子进程后返回子进程号，失败返回-1  Int execve(const char\* path, const char\* argv[], const char\* exvp[])  执行成功后将用一个新的程序代替原进程，但进程号不变，它绝不会再 返回到调用进程了。  Pid\_t wait(int\* status)  Pid\_T waitpid(pid\_t pid, int\* status, int option)  执行成功将返回终止的子进程的进程号，不成功返回-1。  Pid\_t getpid()  返回当前进程的进程号  Pid\_t getppid()  返回当前进程父进程的进程号  Int kill(pid\_t pid, int sig)  kill 发送成功返回接收者的进程号，失败返回-1。  Int pause()  pause 挂起调用它的进程直到有任何信号到达。调用进程不自 定义处理方法，则进行信号的默认处理。只有进程自定义了信号处理方法 捕获并处理了一个信号后，pause 才会返回调进程。  Sighandler\_t signal(int signum, sighandler\_t handler)  signal 调用成功会返回信号处理函数的返回值，不成功返回-1。   1. 执行进程示例程序   如果给定参数的话则子进程负责执行参数，父进程等待子进程的结束。否则父进程不等待子进程。     1. 编写一个多进程并发执行程序。父进 程每隔 3 秒重复建立两个子进程，首先创建的让其执行 ls 命令，之后创建执行让其 执行 ps 命令，并控制 ps 命令总在 ls 命令之前执行。 | | | |
| 结论分析与体会：   1. Fork函数对于父进程返回子进程的pid，对于子进程则返回0，因此可以通过判断pid是否等于0来将父进程和子进程的代码分离。在执行fork之后，子进程会继承fork语句之前父进程所有的信息，在此之后子进程与父进程完全分离。如果父进程在子进程结束前就结束了，那么子进程会变成孤儿进程，挂载到1号进程下。 2. Unix系统使用软中断信号来通知进程发生了异步事件，进程之间通过kill函数来发送软中断。当然内核也会因内部事件而给进程发送信号，例如非法访问，除0等异常。 3. 调用execve函数后，进程不在返回而是直接退出。 4. 当一个正在执行的进程因等待某种事件发生而无法继续执行下去时会从执行状态变为阻塞状态，pause函数完成的就是这个操作，将进程阻塞等待信号的到来。   存疑：在第一个实验中，如果在fork之后父进程执行完毕子进程还没有创建完，则子进程在创建完成后应当挂在到1号进程下，但是实际测试发现子进程挂载到了0号进程。 | | | |